SPECIFICATION FOR
HOUSING MANAGEMENT SYSTEM

Prepared by

Peter Tatian

The Urban Institute
2100 M Street, NW
Washington, DC 20037

UI Project 06283-014
November 1992

Prepared for

INTERNATIONAL CITY/COUNTY MANAGEMENT ASSOCIATION
Local Government and Housing Privatization
USAID Project No. 180-0034

U.S. Agency for International Development, ENI/EEUD/UDH
Contract No. EUR-0034-C-00-2034-00, RFS No. 14

Specification for Housing Management System
9 November 1992

CONTENTS

1 OVERVIEW OF THE SYSTEM

This document contains the complete specification for a housing management system (HMS) to be developed for the United States Agency for International Development Technical Assistance Program in CSFR (RFS #14). This is a fully integrated housing management system that will be made available to municipalities in CSFR and to the companies or entities that are responsible for managing both municipal and private housing.

1.1 Separate programs for municipalities, small owners, and managers

There will be three (3) separate versions of this software developed under this specification -- one for municipalities, one for management companies, and one for small owners. The programs will contain similar functions, but, since municipalities, small owners and managers have different information needs, the programs will differ in the level of detail and the types of functions they provide. For instance, managers will need to keep track of detailed data on housing expenses and revenues, while municipalities only need summaries of this information and will want to compare the performance of the different managers responsible for their properties.

Figure 1.1 summarizes the differences between the two programs. These distinctions will be explained further in the descriptions of each sub-system.

1.2 Program sub-systems

Each of the two programs in the HMS will consist of five main sub-systems. These are:

- A housing database
- An accounting system
- A purchasing and inventory control system
- A maintenance tracking function
- A reference document facility

As was discussed above, the functions provided in each sub-system will be different for the three programs. Each of these sub-systems should be integrated with the others, so that information is exchanged between the different systems and the need for entering the same data more than once is eliminated. The interaction of the sub-systems will be discussed in the subsequent sections of this specification.

Figure 1.1 - Differences between owner and manager programs

Sub-system	Comparison of programs
Housing database	All programs contain the full set of data on buildings, sites, apartments, commercial spaces, and monthly charges. The municipality's program, however, includes a list of housing managers and data on mortgages on the property, while the manager's program includes a list of owners. Reporting capabilities are similar.
Accounting	The manager's program comes with a complete accounting package for recording individual payments and receipts associated with the properties. The municipality's program only deals with monthly financial data aggregated by types of expenditures and revenues. The municipality's program will have a full double-entry accounting package; the small owner's program will only require single entry accounting. The manager's program can produce monthly transfer files, which are read by the owner's program to update the owner's data. Both programs produce financial reports, but the municipality's program has the capability to prepare special reports for comparing the performance of management companies.
Purchasing and inventory	All programs allow record keeping on the equipment in buildings. The municipality's and small owner's programs provide a capital cost estimation function, which reports the remaining useful life of individual building systems and estimates the cost for replacement.
Maintenance tracking	The manager's program and the small owner's program tracks individual maintenance tasks through work orders. The manager can generate lists of unresolved work orders. The municipality's program does not have direct access to work order data, but reads transfer files provided by the manager's program that summarize the monthly total of complaints for different categories and the repair costs and time estimates for each.
Reference documents	An archive of useful documents. Same functions in all programs.

2 HOUSING DATABASE

The first component of the HMS is a database containing information on buildings, sites, apartments, commercial spaces, tenants, owners, and housing managers. Each of these components will consist of separate files that will be linked using key fields. For example, the list of apartments will be linked to the list of buildings through the city code, cadastral no. and description no. (…íslo popis) of the building. The data included in each file is described below.

Unless otherwise indicated, all data are included in all three programs.

2.1 Building data

The following items should be included in the building level data:

- **Ë** Code for city in which building is located*
- **Ë** Cadastral no.*
- **Ë** Building description no. (…íslo popis)
- **Ë** Supplemental description code (see below)
- **!** Name of street
- **!** Street no. of building (…íslo orientacni)
- **!** Site ID
- **!** Type of building (corner, terrace, detached, semi-detached)
- **!** Type of construction*
- **!** Year of construction
- **!** Year of most recent other capital (i.e. major) repair
- **!** Description of most recent capital repair
- **!** Number of floors
- **!** Number of apartments
- **!** Number of commercial spaces
- **!** Total residential floor space (m2)
- **!** Total comm. floor space (m2)
- **!** Total common area floor space (m2)
- **!** Central heating (yes/no)
- **!** Elevator (yes/no)
- **!** Tenure of building (municipally owned, privately owned, coop, etc.)*
- **!** Number of municipally owned units
- **!** Number of privately owned units
- **!** Units individually metered for gas (yes/no)
- **!** Units individually metered for electricity (yes/no)
- **!** Units individually metered for water (yes/no)
- **!** Name of manager assigned to building
- **!** Does manager reside in building (yes/no)
- **!** National historic site (yes/no)
- **!** Appraised value of property
- **!** Date of appraisal
- **!** Date of purchase of property
- **!** Purchase price

! Active mortgage on property (yes/no)
! Bank account number
! Name of bank
" Date when information last updated and user ID
! Memo

In this file (and in the descriptions of subsequent files) the following symbols will be used:

Ë = Key field (these variables uniquely identify each file record)
! = Field entered by the user
* = Variable values are user-defined codes
" = Not entered by the user, but filled in automatically by the software

In the building data, the first four (4) items are the key fields that uniquely identify each building in the database. All key fields should be stored as character data, so that letters and symbols may be entered as well as numbers. The supplemental description code is a one character field to deal with cases where two buildings in the same cadastre have the same description number. The program should alert the user if two buildings have the same values for the key fields.

The items with an asterisk (*) indicate **user-defined codes**. That is, the program should allow the user to store a list of coded responses for each of these items. For example, the list of codes for "type of construction" could include: panel, brick, stone, wood, etc. The user should be able to modify these lists so that new responses may be added or existing responses changed or deleted. (In all subsequent file descriptions, the asterisk will indicate user-defined codes.)

"Memo" is a field allowing the user to attach a note of indeterminate length to the building record. Whenever information in a building record is changed, the program should replace the "date when information last updated" with the current date and "user ID" with the user identification name of the person who made the changes.

2.2 Site data

The following information should be included on the building sites (there will be one site record for every building):

Ë City code*
Ë Cadastral no.*
Ë Building description no.
Ë Supplemental description code
! Number of buildings on site
! Tenure of site (owned, leased)*
! Size of site (m2)
! Site reference number (character field)
! Water connection (yes/no)
! Electricity connection (yes/no)
! Sewer connection (yes/no)

! Appraised value of the site
! Date of appraisal
! Date of purchase of site
! Purchase price
! Active mortgage on site (yes/no)
" Date when information last updated and user ID
! Memo

The site reference number is a character field containing an externally defined ID number (used to refer to magistrate records, for instance).

2.3 Apartment data

2.3.1 Basic data

The database should include the following information on all residential units:

Ë City code*
Ë Cadastral no.*
Ë Building description no. (…íslo pop.)
Ë Supplemental building description code
Ë Unit no.[1]
! Floor on which apt. is located
! Category (I, II, III, IV)
! Occupancy status (occupied, vacant, being renovated)
! Tenure of unit (rented, condominium, coop, etc.)*
! Floor space (m2)
! Number of rooms
! Bathroom (full, half, common outside apt., none)
! Toilet (inside apt., common outside apt., none)
! Bathtub (yes/no)
! Kitchen (yes/no)
! Stove (yes/no)
! Oven (yes/no)
! Refrigerator (yes/no)
! Cellar storage (yes/no)
! Type of heating*
! Hot water source*
! Type of floor covering*
! Rent calculated from eviden…ni list (yes/no)

"Unit no." is used to refer to the ID numbers assigned to both residential and commercial spaces. It is assumed that residential and commercial units are numbered uniquely within a building, that is, no residential unit has the same number as any other residential or commercial unit, and no commercial unit has the same number as any other commercial or residential unit.

! Monthly clear rent
! Apartment reserved (yes/no)
! Date when apt. will be ready for new occupants
" Date when information last updated and user ID
! Memo

2.3.2 Eviden…ní list

In addition to the above information, a separate file should include any additional information needed for the rent calculation for 1993 (eviden…ní list). The user should have the option of entering the rent directly, or asking the program to calculate the rent from the eviden…ní list.

2.4 Commercial space data

2.4.1 Basic data

The database should include the following information on all commercial spaces:

Ë City code*
Ë Cadastral no.*
Ë Building description no. (…íslo pop.)
Ë Supplemental building description code
Ë Unit no.[1]
! Floor on which unit is located
! Occupancy status (occupied, vacant)
! Tenure of unit (rented, condominium, coop, etc.)*
! Floor space (m2)
! Number of rooms
! Bathroom (full, half, common outside apt., none)
! Toilet (full, half, common outside apt., none)
! Bathtub (yes/no)
! Kitchen (yes/no)
! Stove (yes/no)
! Oven (yes/no)
! Refrigerator (yes/no)
! Cellar storage (yes/no)
! Type of heating*
! Hot water source*
! Type of floor covering*
! Rent calculated from eviden…ni list (yes/no)
! Monthly clear rent
! Unit reserved (yes/no)
! Date when unit will be ready for new occupants
" Date when information last updated and user ID
! Memo

2.4.2 Eviden…ní list

In addition to the above information, a separate file should include any additional information needed for the rent calculation for 1993 (eviden…ní list). The user should have the option of entering the rent directly, or asking the program to calculate the rent from the eviden…ní list.

2.5 Residential tenant data

2.5.1 Primary tenant and lease data

The following information on the primary tenants and the lease terms should be included in the database (note: this is an apartment-level file, so there is only one record per apartment):

- **Ë** City code*
- **Ë** Cadastral no.*
- **Ë** Building description no. (…íslo pop.)
- **Ë** Supplemental building description code
- **Ë** Unit no.
- **!** Family name (primary tenant)
- **!** First name (primary tenant)
- **!** Number of inhabitants
- **!** Number of adults
- **!** Number of children
- **!** Telephone number of apt.
- **!** Employment of primary tenant*
- **!** Name of employer
- **!** Address of employment
- **!** Date of start of occupancy/vacancy
- **!** Lease number
- **!** Length of current lease (months)
- **!** Date current lease expires
- **!** Amount of notice required to terminate lease (days)
- **!** Date notice given
- **!** Move-out date
- **!** Amount of security deposit (kcs)
- **!** Deposit of last month's rent (kcs)
- **!** Telephone number of employment
- **"** Date when information last updated and user ID
- **!** Memo

2.5.2 Data on all occupants

This section contains data on all occupants in the apartment (including primary tenant):

- **Ë** City code*
- **Ë** Cadastral no.*
- **Ë** Building description no. (…íslo pop.)
- **Ë** Supplemental building description code
- **Ë** Unit no.

! Family name
! First name
! Date of birth
! Sex (M/F)
! Relation to primary tenant (wife, child, etc.)*
! Birth no.

Certain fields in 2.5.1 (number of inhabitants, adults, and children) should be updated automatically whenever changes are made in 2.5.2.

2.6 Commercial tenant data

The following information on commercial tenants should be included in the database. There may be more than one tenant per commercial unit:

Ë City code*
Ë Cadastral no.*
Ë Building description no. (…íslo pop.)
Ë Supplemental building description code
Ë Unit no.
! Name of company or organization
! I„ O (Organization ID number)
! Type of company or organization*
! Use of comm. space*
! Family name of representative
! First name of representative
! Telephone number of company or organization
! Fax number
! Home address of representative
! Home telephone number
! Protected tenant - cultural organization, school, etc. (yes/no)
! Date of start of occupancy/vacancy
! Lease number
! Length of current lease (months)
! Date current lease expires
! Amount of notice required to terminate lease (days)
! Date notice given
! Move-out date
! Amount of security deposit (kcs)
! Deposit of last month's rent (kcs)
" Date when information last updated and user ID
! Memo

2.7 Other charges

The existing functions in the Meson housing management software for assessing charges

to tenant accounts should be retained in the new software. The user should be able to customize this function by defining as many different types of charges as he likes.

2.8 Owner data [manger's program only]

The database should include the following information on property owners:

- Owner ID no.
- Type of owner (municipality, bank, individual)*
- Name of owner
- Address
- Telephone no. (day)
- Telephone no. (evening)
- Fax no.
- Date when information last updated and user ID
- Memo

Each owner should be listed only once in this file. Since the links between the owner and building data are complex, they are described in detail in §2.11.

The list of owners is included only in the manager's program.

2.9 Management company data [municipality's program only]

The database should include the following information on housing management companies:

- Management company ID no.
- Name of management company
- Address
- Telephone no. (day)
- Telephone no. (evening)
- Fax no.
- Name of company owner
- Statutory representative for firm
- Date when information last updated and user ID
- Memo

Each management company should be listed only once in this file. Since the links between the manager and building data are complex, they are described in detail in §2.11.

The list of management companies is included only in the municipality's program.

2.10 Mortgage data [municipality's and small owner's program only]

The database should allow the owner to record information on mortgages taken out against

the property or site. There may be more than one mortgage listed under a site:

- Ë City code*
- Ë Cadastral no.*
- Ë Building description no. (…íslo pop.)
- Ë Supplemental building description code
- Ë Number of mortgage (1st, 2nd, 3rd, etc.)
- ! Name of lending institution
- ! Address
- ! Phone number
- ! Mortgage taken out on: property, site, property and site
- ! Type (fixed-rate, variable rate, dual-index, other)
- ! Date of mortgage
- ! Mortgage reference number (character field)
- ! Loan amount
- ! Current annual interest rate
- ! Term of loan (months)
- ! Monthly payment
- " Date when information last updated and user ID
- ! Memo

2.11 Links between data files

Figure 2.1 shows the connections between the different data files described above. Files are linked by their key fields. Special considerations regarding file linking and exchange of data between files is given below.

2.11.1 Site and building data

It should be noted that a single site may contain more than one building. In this case, the site would be entered more than once in 2.2. The building-level records contain the site ID, which links the building data to the site data. (NB: It is assumed that a building may not be on more than one site.)

2.11.2 Apartment/comm. space and building data

The apartment and commercial space data files are linked through the city code, cadastral no., and the building description no. The program should update the following fields in the building-level data automatically whenever the apartment or commercial space data are changed:

- ! Number of apartments
- ! Number of commercial spaces
- ! Total residential floor space (m2)
- ! Total comm. floor space (m2)

The program should prompt to user before updating these items, to give the user the opportunity to override this function.

2.11.3 Rent calculation

Any changes in building, apartment, or comm. space data that would result in a change in the rent calculation should cause the program to ask the user if he wishes to recalculate the rent amounts. The new rent amount would be included in the apt. or comm. space data (2.3 or 2.4).

2.11.4 Owner and building data [manager's program only]

The links between owner and building data are complex, since a single owner may possess several buildings, and a building may belong to more than one owner. The owner data (2.8) defines the list of property owners. The following linking file should be used to connect the owner and building lists:

Ë City code*
Ë Cadastral no.*
Ë Building description no. (…íslo pop.)
Ë Supplemental description code
! Owner ID
! Owner's share in building

More than one owner may be listed for a single building. The file connects to the owner list through the owner ID. The "owner's share" is a character field, which allows the user to enter information in fraction form, like "4/10" ("four-tenths").

2.11.5 Management company and building data [municipality's program only]

A similar file is used to link the management company and building data for the municipality's program. In this case, however, there is only one management company per building.

Ë City code*
Ë Cadastral no.*
Ë Building description no. (…íslo pop.)
Ë Supplemental description code
! Management company ID

The file connects to the management company list (2.9) through the management company ID.

2.12 Waiting list

The database should keep the following set of information on applicants waiting for an apartment to become available in a building:

Ë City code*
Ë Cadastral no.*
Ë Building description no. (…íslo pop.)
Ë Supplemental description code

Ë Number of applicant on list
! Name of applicant
! Date of birth
! Birth number
! Date original application made
! Type of apartment desired (efficiency, one bedroom, etc.)
! Home telephone number
! Home address
! Employment
! Name of employer
! Employment address
! Employment telephone number
! Length of employment
! Annual wages
! Date when applicant information last updated
! Date unit assigned
! Unit number assigned
! Move-in date
! Memo

2.13 Database functions

The database sub-system will, of course, contain functions for entering and editing the data described above. The program should include a number of "user-friendly" features to facilitate the management of the database.

The database subsystem should contain the following functions (2.13.1 - 8):

2.13.1 Edit data

The user should be able to edit all of the data files described above. The data entry forms should contain understandable descriptions of each item, not just variable names. For those items with user-defined codes (*'d items), the description of the code should be displayed, and not the code itself. For example, for type of building construction, the field would display "Panel" and not the numeric code for a panel building.

When entering data for user-defined fields, the user should be able to pull up a menu of the previously defined codes (by, say, pressing a special function key) and choose a response from the list. The program would then automatically insert the appropriate code into the database.

The program should include data checking where appropriate to prevent the user from entering invalid data. Key fields should be verified for uniqueness (that is, no two records in a file should have exactly the same values for the key fields).

2.13.2 Add a new building

When entering data on a new building, the program should automatically prompt the user

to enter data into all the relevant files. For example, the user should first enter the building-level data (2.1), and then proceed to the site data (2.2). In the case of the municipality's version of the program, the user would then enter the ID number of the management company responsible for the building (2.11.5). This ID would either be selected from the list of management companies (2.9), or, if the company is not in the list, the user would be allowed to add the company to the list. Similarly, for the manager's program the user would enter the ID numbers of the owners of the buildings (2.11.4). If necessary, the user could add new owners to the list (2.8).

Once the building, site, and owner or manager data have been entered, the user would then be taken automatically to the apartment data (2.3). Once all the apartments have been entered, the program asks the user to enter data on commercial spaces (2.4). After entering all of the data for a building, the user is asked if he wishes to repeat the process for another building. During the data entry process, the program should allow the user to skip over any of the data files.

When editing existing data, the user should be able to search for buildings by city, cadastral no., and description no., or by city and street address. Once a building has been selected, the user should be able to call up any of the data associated with that building (building, site, apartment, comm. space, owners/manager) by selecting a command from a menu or pressing a function key. The user should be able to jump between the different data files at will. When editing apartment or comm. space data, the user should be able to search for a unit by its ID no. or by the name of the tenant in the unit.

2.13.3 Delete a building from the database

This function permits the user to delete a building and all associated data from the database. The program should confirm this operation before proceeding.

2.13.4 Add an apartment or commercial space

As with the building data, the user should be prompted to enter data into all relevant unit- and tenant-level files.

2.13.5 Delete an apartment or commercial space

This function deletes all data for the chosen apartment or commercial space. The user should confirm the operation before the program proceeds.

2.13.6 Move-in a new tenant

This function allows the user to move a new tenant into a vacant apartment (the function should not permit moving a tenant into an occupied apartment). If the new tenant is on the waiting list, the program should automatically copy the available information from §2.12 to the tenant information file §2.5.1. It should then prompt the user to enter the remaining tenant data. If the new tenant is not on the waiting list, the user is prompted to enter all relevant data to add the future tenant to the waiting list.

2.13.7 Move-out a tenant

The user executes this function when an existing tenant moves out of his apartment. All relevant data are deleted and the occupancy status of the unit is set to vacant. The function should also generate a **report** that gives the account balance of the tenant, that is, any outstanding charges that have yet to be paid. (This information will come from the accounting data in §3.) The data on the former tenant is then transferred to §2.14.

2.14 Former tenants

This section keeps data on former tenants who have left the building. The most important use of this information is to keep records on outstanding charges owed by former tenants:

- **Ë** City code*
- **Ë** Cadastral no.*
- **Ë** Building description no. (…íslo pop.)
- **Ë** Supplemental building description code
- **Ë** Unit no.
- **!** Residential or commercial tenant
- **!** Name of tenant, company or organization
- **!** Telephone number
- **!** Fax number
- **!** I„ O (Organization ID number)
- **!** Name of employer
- **!** Address of employment
- **!** Telephone number of employer
- **!** Move-in date
- **!** Move-out date
- **!** Lease number
- **!** Amount of deposits refunded
- **!** Date deposits refunded
- **!** Amount of rent charges left to pay
- **!** Amount of other charges left to pay
- **!** Date of last payment
- **!** Amount of last payment
- **"** Date when information last updated and user ID
- **!** Memo

2.15 Reports

The database sub-system should be capable of producing the following set of reports:

Building/site passport: By this we do not mean the traditional OPBH passport, but rather a one page printout with all of the building- and site-level data described in sections 2.1 & 2.2. In addition, the building passport for the municipality's program should include the information on the building manager (2.9), while the passport for the manager's program should include the list of building owners (2.8) and their shares in the building (2.11.4).

Apartment/tenant passport: Again, this is a one page printout with the apartment-level data included in sections 2.3.1 & 2.5, excluding the eviden…ní list.

Comm. space/tenant passport: A one-page printout with the comm. space data given in §2.4.1 & 2.6.

Eviden…ní list: This report contains the complete eviden…ní list for an apartment (2.3.2) or a commercial space (2.4.2).

List of buildings: A listing of one or more buildings, selected by criteria (see below). The report contains one line per building that gives variables selected by the user.

List of apartments/commercial spaces: A listing of all apartments & commercial spaces in a selected set of buildings. A header with the building's identifying data would be followed by a list of apartments (with apt. no. , tenant's name, and any other data chosen by the user). A similar report should be available for commercial spaces.

List of owners: This report is only available in the manager's program. It is a listing of all owner data described in §2.8.

List of management companies: This report is only available in the municipality's program. It is a listing of all management co. data described in §2.9.

Vacancy report: Presents separate lists of all units in a building that (1) are currently vacant, (2) will become vacant in the next 30 days, (3) units whose leases expire in the next 30 days. For (1), the report gives the date that the unit became vacant (date of occupancy/vacancy from §2.5.1 or 2.6.1), the number of days it has been vacant, the loss in rent from the vacancy, the date when the apartment will be ready for occupancy, and the move-in date for a new tenant (if any). For (2), the report gives the move-out date (from §2.5.1 or 2.6.1), the monthly clear rent, the move-in date for a new tenant (if any). For (3), the report gives the lease expiration date, the move-out date (if the tenant is not renewing the lease), and the monthly clear rent.

Schedule of move-ins: This report shows the scheduled move-in dates of all applicants assigned units (from §2.12).

Waiting list: This report gives lists of all persons on the waiting list separated by the type of apartment desired. The listing should contain the name of the applicants, date of original application, home telephone number, employment telephone number, date unit assigned, unit number assigned, and the move-in date.

Former tenant list: This report lists all the former tenants from §2.14, the date they moved out, the amounts of rent and other charges they still owe, and the date and amount of the last payment made.

For all of the report's described above, the user should be able to select either a single building, all buildings, or a set of buildings to be included in the report. For example, a housing

manager may wish to print out the passports for all buildings owned by Prague 2.

Alternatively, the user should be able to select an individual building by searching through a list sorted by city, cadastre, and description no., or by city and street address.

Finally, the user should be able to create reports for all buildings in the database.

3 ACCOUNTING SYSTEM

The second major sub-system of the HMS is an accounting program that can be used to keep track of all revenues and expenses associated with housing. For the municipality's and manager's programs, the accounting system should be based on the double-entry method of accounting and should conform to standard accounting rules that will be in effect in CSFR on 1 January 1993 for municipal governments and private firms, respectively. For the small owner program, a simpler single-entry accounting system should be provided. Financial accounting for each building must be functionally separate, that is, a separate "set of books" is kept for each building.

All three programs will use the same base information. However, the programs will have different ways of accessing this data. The differences between the programs will be discussed in §3.14.

The following sections describe the main functions of the accounting sub-system. Any further analysis needed on this topic will be carried out by Meson and implemented into the system.

3.1 Chart of accounts

The chart of accounts is used to track different categories of expenditures and receipts. The chart of accounts list should include the following information:

- Account code
- Description

The user should be able to add, delete, or edit the chart of accounts from this function. The user should also be able to identify the account codes for rents and late fees, as these are special cases for the software.

3.2 Vendor master

The *vendor master* is a list of each vendor or service provider used by the management company (plumbers, electric company, hardware suppliers, etc.). The vendor master should include the following:

- Vendor ID no.
- Name of vendor
- Type of service*
- Address
- Telephone no.
- Fax no.
- Chart of accounts code (from §3.1)

The chart of accounts code is entered only if the vendor provides a single service (for example, plumbing). The code tells the program which account should be credited whenever

payments are made to the vendor. For example, paying the plumber should result in a credit being made to "Repairs - Contract."

3.3 Payroll

The accounting system should also be used to record the salaries of employees who work on a building. The following information on employees should be included:

- **Ë** City code*
- **Ë** Cadastral no.*
- **Ë** Building description no. (…íslo pop.)
- **Ë** Supplemental description code
- **Ë** Employee ID number (internal)
- **!** Family name
- **!** First name
- **!** Employee social security number [Olga - translate as appropriate to CSFR]
- **!** Type of employee (full-time, part-time)*
- **!** Position (technician, janitor, etc.)*
- **!** Date of birth
- **!** Sex (male/female)
- **!** Home address
- **!** Home telephone
- **!** Payment period (monthly, bi-weekly, weekly, hourly, other)*
- **!** Percentage of salary charged to this building
- **!** Amount paid per period
- **!** Amount of social security payment
- **!** Child allowance
- **!** Sickness payments
- **!** State grants
- **!** Current vacation hours
- **!** Vacation hours used - year to date
- **!** Current sick hours
- **!** Sick hours used - year to date
- **!** Memo

3.4 Receipts journal

3.4.1 Recording payments

This section is used to record payments from tenants received as cash, postal checks, bank account transfers, or other means. Payments are recorded in a **receipts journal**. The receipts journal is designed to provide a convenient way of entering payment data. (It should be noted, however, that certain payments will be recorded through the general ledger.) The receipts journal for a tenant summarizes the charges past due from previous months, the current month's charges, and payments received during the current month:

- Ë City code*
- Ë Cadastral no.*
- Ë Building description no. (…íslo pop.)
- Ë Supplemental building description code
- Ë Unit no.
- Ë Chart of accounts code for charge
- ! Past due charges from previous months
- ! Current month's charges
- ! Current month's payments
- ! Balance due

The first five (5) items identify the tenant by building and unit no. The account code describes the type of charge (as defined in the chart of accounts, §3.1). The next item gives the amount of this charge that is past due from previous months. Also included are the charges and payments for the current month. Finally, the balance due indicates the amount owned by the tenant, and is equal to the past due charges, plus the current charges, minus current month's payments.

The receipts journal entries for every tenant are updated at the start of the month as part of the end of month closing (§3.12). The current month's payments and charges are zeroed and the past due charges are set to the balance due. Then, the program adds each tenant's rent (from 2.3 or 2.4) and other regular charges (2.7) to the current month's charges and balance due. The program should debit any unpaid charges from the previous month to accounts receivable, and credit the same amount to the income account. In addition, it must debit the rental income account by the total amount of the gross potential rent (i.e. the rent that would be received if 100 percent of the tenants paid their monthly rent in full).

When the user wishes to record payments received from a tenant, he first finds that tenant's building (by searching a list of buildings) and unit no. (by searching a list of units). The program then displays a receipts journal form like that in Figure 3.1.

Figure 3.1 - Receipts Journal

Receipts Journal for July 1992

Tenant: TAJ „ MAN Petr, Apt. 6, 1 CS Armady, 160 00 Praha 6
Dejvice, Cis. pop. 123

Date: 15 July 1992

Type of charge	Past due	Current month charges	Current month payments	Balance due	Today's payments
Rent	250	250	0	500	.
Electricity	0	120	0	120	.
Heat	0	10	0	10	.
Water	0	60	0	60	.
TOTAL	250	440	0	690	0.00

Type of payment: Payment ID no.:

When payments are received from the tenant, they are entered in the last column (today's payments). The type of payment (cash, postal check, etc.) and the ID number of the payment (for postal checks or bank drafts) should also be entered. Once all payments have been entered, the program adds them up and displays the total in the last box. The user is asked to verify that the total is correct. If not, the user may make corrections to the payments.

Pressing a special function key should cause the amounts shown in "Balance due" to be entered automatically in the payments column, providing a quick way of indicating that the tenant has paid his bill in full. The program should also permit "pre-payments," that is, payments larger than the balance due, but it should ask the user to confirm any such payments before accepting them.

The program should be able to print a summary of the day's transactions, so that the user can verify the amounts entered. The program should also be able to print up a bank deposit slip, which lists the items on a payment by payment basis. Separate bank slips should be generated for each bank account where money should be deposited (different owner's accounts, for instance). When the deposit slip is printed, the appropriate general ledger asset (cash) account should be debited automatically (this will balance the ledger). If a payment is to be made to a vendor, then the vendor information (§3.2) should be included on the bank deposit slip.

The program should also be able to print **payment receipts** with the name and address of the tenant, the date, the charge balance before the payment, the amount of the payment received, and the remaining account balance. The user should be able to generate receipts for each payment

as it is entered, or for all payments entered that day.

During the end of day closing, the program automatically makes appropriate entries in the general ledger for the payments entered into the receipts journal, that is, it credits the income account ("Rental income") and debits the cash account (usually "Cash in bank - operating account"). If the payment was for past due rent, however, the accounts receivable account would be credited, while the cash in bank account would be debited. (A separate menu function should allow the user to specify the exact codes for accounts to be debited and credited.)

The next time the receipts journal is displayed for this tenant, it will reflect the updated payment information. The program should be able to print a list of payments and charges for an individual tenant, or for all tenants in a building.

3.4.2 Adding charges

The receipts journal should also allow the user to post additional charges to the tenant's account at any time. In addition, the program should be able to charge late fees automatically. For example, for rental charges over 30 days past due, the program could assess a late fee of 5 percent. Late fees would appear as a separate row on the receipts journal, and would therefore also have a separate code in the chart of accounts. The program should allow for different late fee policies for each building.

3.4.3 Corrections

If an error is discovered in a previous entry, the program should allow for a correction to be made by means of a **contra entry**. For example, on 5 July a rent payment of 100 kcs is posted to Mr. Taj...man's account. The next day, the operator discovers that this was incorrect. The user can call up Mr. Taj...man's record and enter a contra entry of (100) kcs, which will result in a cancellation of the previous entry. The correct entry may then be made. (NB: All transactions already posted remain in the general ledger.)

3.4.4 Exchanging payment data on magnetic media

The program should provide functions for receiving automated payment data on a diskette from the Rental Collection Center and the Post Office, as well as sending updated rent amounts to the Rental Collection Center on diskette.

3.5 Accounts receivable

The software should have the capacity to generate tenant accounts receivable reports. There would be two formats for these reports:

Delinquency list by tenant: This is a listing of each tenant in a building who has either a debit or credit balance. The list would give the unit number, tenant family name, the debit or credit balance for rent, the balance for all other charges, and the total balance.

Aged accounts receivable: This is similar to the first report, but the delinquencies are

divided into periods of 0-30, 31-60, 61-90, and over 90 days.

3.6 Expense journal

As in the case of cash receipts, the chart of accounts will have categories for all types of administrative, operating, utility, repairs and maintenance, taxes, insurance, and financial expenses.

Expenses should be posted as received to an **expense journal**. The user should be able to access this journal and post expenses to the appropriate expense accounts. The information entered should include: date of posting, date of bill, date payment due, period covered (if applicable), vendor ID, a brief description, amount of the bill, expense account codes (the account codes should be filled in automatically if they are included in the vendor data, see 3.2).

There should also be a "pay or hold" function in the expense journal. If the bill is to be paid immediately, then the journal entry should be marked "pay." If the bill is to be held for any reason, such as the receipt of additional revenue, then the entry would be marked "hold."

For bills to be paid immediately, a separate function should print checks or bank drafts, debit the appropriate expense account, and credit the cash account.

If the bill is to be held, the appropriate expense accounts should be debited and the accounts payable account should be credited. When held bills are paid, the accounts payable account is debited and the cash account is credited.

As was the case with the receipts journal, it should also be possible to revise the expense journal through contra entries.

3.7 Accounts payable

The accounts payable function allows the user to get a **report** of all bills that need to be paid and to instruct the program to make payments. At any time, or at least on a weekly or monthly basis, the user should generate a list of bills that need to be paid. This list should be in chronological order (by the date the payment is due) or alphabetically by vendor.

If there is enough cash available, the user can order the system to "pay" items from the accounts payable listing. At that time, the accounts payable account is debited and the cash account is credited.

The accounts payable report should also be available on a per vendor basis. This report would give both the charges and payments made to each vendor.

The user should also be able to generate a cumulative listing of bills paid to date for the month (**the "month-to-date" check register**).

3.8 General ledger

The general ledger is used to record all financial transactions in the standard format for a double-entry accounting system, with asset, liability, capital, income, and expense accounts. The general ledger can be updated either daily, as the journals are closed each day, or monthly. Accounting is done on an **accrual basis**, that is, accounts payable and receivable are updated monthly, and the expense accounts reflect expenses incurred and not just what was actually disbursed.

The following information is included in the general ledger:

- **Ë** City code*
- **Ë** Cadastral no.*
- **Ë** Building description no. (…íslo pop.)
- **Ë** Supplemental description code
- **Ë** Transaction no. (internally generated sequential number)
- **"** Date of posting
- **!** Chart of accounts code
- **!** Description of transaction (character field)
- **!** Debit amount
- **!** Credit amount
- **!** Unit no.
- **!** Work order no. (see §5.1)
- **!** Vendor ID (if applicable)
- **"** ID of user posting transaction
- **!** Type of payment* (cash, postal check, bank transfer, etc.)
- **!** Payment reference number (number of postal check, etc.)
- **!** Memo

3.8.1 Making journal entries

The journal entry function allows the user to post transactions not handled by the receipts or expense journals to the general ledger. This includes posting interest to cash accounts, corrections to general ledger, and changes in the equity accounts.

When making a journal entry, the user enters all of the information described in the general ledger. The transaction no. and date of posting are filled in automatically by the program. While entering transactions, the user should be able to call up the chart of accounts to find the appropriate account codes. The unit no. is entered only for transactions involving a specific unit (e.g. payment of rent). Note that when payments are recorded through the receipts journal or when charges are recorded in the expense journal, the program should copy the building and unit information automatically into the general ledger. Similarly, the work order no. is entered if the transaction is a result of a specific work order.

By including the unit no. and work order no. in the general ledger entries, the program can track revenues and expenses on a unit and work order basis, and can link the accounting and other sub-systems. For example, entering a transaction in the general ledger may result in information

in other parts of the program being changed. The receipt of rent payments recorded in the ledger should cause the receipts journal (3.4) to be updated. Payments made because of a work order would be added to the work order data (5.1). These updates will occur automatically without any prompting from the user, though the program should inform the user that they are taking place.

At any time the user should be able to view or **print a list of journal entries** for a specified period of time.

3.8.2 Corrections to the general ledger

Corrections to the receipts and expense journals should be made through those menu items, as explained in sections 3.4 and 3.6. There should also be a correction function (accessed through the menu or a function key) for the general ledger. The correction function allows incorrect entries to be canceled through a contra entry. By entering a negative amount, for example: (1000) kcs, the user may cancel a previous entry that was incorrect. The correct amount can then be entered. (Both the original transaction and the correction remain in the general ledger.)

3.9 Monthly trial balance

The monthly trial balance function tells the user whether or not the books "balanced," that is, if the total amount of debits and credits are equal. The **trial balance report** lists every account and gives the total debits and credits for each for the current month. The total debits for all accounts should be equal to the total credits.

3.10 End of day closing

At the end of the day, the system should post to the general ledger all of the day's receipts entered in the receipts journal (as described in §3.4) and payments entered in the expense journal (§3.6). It should print a list of all receipts and payments for the day, as was described above. In addition, any payments received but not deposited should be recorded in accounts receivable and undeposited receipts.

3.11 End of month closing

At the end of the month, the user should "close the books." This function should generate the following reports:

Check register - This report gives the cash account balance at the beginning of the month, a list of all payments made from the cash account and receipts entered into the cash account during the month, and the current cash balance.

Rent roll - This report lists every tenant on a building-by-building basis and indicates: the date the tenant's lease expires, the monthly clear rent, the previous month's outstanding balance, the total rent and other charges billed for the month, total payments received for rent and other charges, any credits to the tenant's account, and the new balance at the end

of the month.

As was described in §3.4, the receipts journal and appropriate income and receivable accounts should be updated at this time. The program should also generate bank deposit slips for all undeposited receipts.

3.12 End of year closing

A function should be provided for closing the books at the end of the year.

3.13 Reports

The accounting sub-system should include a reports generator that will produce the following types reports on a building-by-building basis:

- Accounts receivable
- Accounts payable
- Month-to-date check register
- Cash flow report
- Profit and loss statement
- Rent-roll (described above)

The profit and loss reports should give month-to-date and year-to-date accruals, and present budget versus actual income and expenses. These reports should be available for individual buildings, or for groups of buildings.

3.14 Differences between the programs

As was mentioned at the beginning of this section, the accounting functions will be different for the municipality's and manager's programs. The manager's program will have the full set of accounting functions described above. The municipality's program, however, will only require a more limited set of functions.

The manager's program should have a special function for creating transfer data sets for updating the municipality's information. These update data sets will contain aggregated monthly totals type for each building belonging to the municipality. There will be separate totals in the transfer data files for each type of expenditure or receipt. The municipality's program will have a function for updating its data using these transfer data files. (When implementing these functions, it should be remembered that a municipality may be using more than one manager, and a manager may be working for more than one owner.)

Since the municipality does not need to edit individual transactions, it does not need the entry and editing functions in the cash receipts, accounts receivable, expenses, accounts payable, general ledger, journal entries, corrections, trial balance, and closing functions. It should, however, be able to produce all of the reports described above, including:

- Receipts journal for a tenant
- Cash flow report for a building or group of buildings
- Profit and loss statement for a building or group of buildings

All of these reports would be available on a monthly or yearly basis. The owner should also be able to generate the above reports aggregated for all buildings managed by a particular management company. This will allow the owner to compare the performance of management companies.

3.15 Export to Gordic accounting system

Because municipalities in Prague are required to use Gordic accounting software, the accounting system provided with the HMS should be able to export data into a format that may be read by Gordic programs.

4 PURCHASING AND INVENTORY CONTROL

The HMS should include a system for purchasing and inventory control, which would allow the manager and owner to keep track of supplies and equipment in the buildings. The user will set up codes for materials and equipment, grouping them into separate categories and sub-categories. There will actually be two separate inventory files, one for equipment in buildings and one for general supplies. The sub-system should also contain a function for estimating capital repairs to buildings.

4.1 Building equipment inventory

This section contains data on the major physical components or pieces of equipment for the buildings, including elements of the heating, plumbing, and electrical system, clothing washers and dryers, air conditioners, refrigerators, the building roof and facade, etc. The building inventory file should contain the following information:

- City code*
- Cadastral no.*
- Building description no. (…íslo pop.)
- Supplemental building description code
- Unit no. (if applicable)
- Type of equipment*
- Type of material (asphalt, concrete, etc.)*
- Quantity
- Unit of measure (pieces, m2, m, etc.)*
- Model, make or type (character field)
- Serial number (character field)
- Year of construction or installation
- Purchase price per unit
- Condition*
- Replacement cost per unit
- Date of replacement cost estimate
- Estimated useful life (years)
- Owned by (1=Management co., 2=Bldg owner, 3=Other)
- Memo

The type of equipment is a user-defined code indicating the item described for each data record. The program should permit the user to omit the unit no., indicating that the equipment is not associated with a particular unit in a building.

The program should be able to produce a list of equipment for a single building, selected buildings, or all buildings.

4.2 Supply inventory

The general supply file is used to keep an inventory of supplies for each building, such as plumbing supplies, paint, light bulbs, etc. It contains the following information:

Ë City code*
Ë Cadastral no.*
Ë Building description no. (…íslo pop.)
Ë Supplemental building description code
Ë Type of supply*
Ë Lot number (internally generated sequential number)
! Quantity
! Unit of measure*
! Model, make, or type (character field)
! Date of purchase
! Purchase price per unit
! Estimated useful life
! Memo

Type of supply is a user-defined code that identifies the item in stock.

The program should be able to produce **a list of supplies** for selected buildings.

4.3 Purchase orders

When supplies or equipment are purchased, they should be added to the inventory. Supplies and equipment are acquired by issuing a purchase order, which authorizes the expenditure of money for a purchase. The purchase order is similar to the receipts journal (§3.4), but deals specifically with physical items as opposed to heat, water, services, etc.

4.3.1 Issue a purchase order

Each purchase order should contain the following information:

Ë City code*
Ë Cadastral no.*
Ë Building description no. (…íslo pop.)
Ë Supplemental building description code
Ë Unit no.
Ë Purchase order number (system-generated sequential number)
! Date of purchase order
! Vendor ID (from §3.2)
! Chart of accounts code
! Cost of items ordered
! Amount of tax
! Total cost
! Pay/hold
! Payment due date
! Date paid
! Memo
" ID of user entering purchase order

In addition, the following information is recorded for each item to be purchased:

- Ë City code*
- Ë Cadastral no.*
- Ë Building description no. (…íslo pop.)
- Ë Supplemental building description code
- Ë Unit no.
- Ë Purchase order number
- Ë Item number (system-generated sequential number)
- ! Type of equipment/supply*
- ! Quantity
- ! Unit of measure (pieces, m2, m, etc.)*
- ! Model, make or type (character field)
- ! Purchase price per unit
- ! Estimated useful life (years)
- ! Date item received

The program should print out each purchase order once it is prepared. As with the expense journal, the pay/hold field indicates whether the purchase order is to be paid immediately or whether the payment is to be held for a later date. If the purchase order is to be paid immediately, the program should debit the chart of accounts code indicated on the purchase order, and credit the cash account. If the purchase order is to be held, the program should debit accounts payable and credit the cash account. Once a held purchase order is paid through the accounts payable function (§3.7), the program debits the chart of accounts code and credits accounts payable.

The "date item received" field is only entered once the item has been delivered (see below).

4.3.2 Recording receipt of ordered items

The receipt of ordered items must be recorded in the purchase order and inventory information. When an item is received, the date of receipt is entered on the purchase order and the new items are added to the inventories (§4.1 or 4.2). If all items ordered are received at once, a special function should allow the user to record the receipt of all items simultaneously.

4.3.3 Printing out purchase orders

The sub-system should be capable of printing individual purchase orders, or producing lists of purchases orders meeting the following criteria: (1) all purchase orders to a specific vendor, (2) all purchase orders issued over the specified period of time, (3) all purchase orders where the items have not yet been received.

4.4 Capital repair [municipality's and small owner's program only]

An additional report that the municipality's and small owner's program should generate would help estimate the future costs for capital repairs. This report would list all the equipment in a building (from §4.1) and give the following information:

- Type of equipment
- Type of material
- Model, make, or type
- Estimated useful life
- Year of installation or construction
- Original cost
- Number of years estimated useful life remaining
- Predicted replacement cost

The predicted replacement cost would be estimated by multiplying the original cost by an inflationary factor provided by the user.

4.5 Differences between the programs

As with the financial data, the manager's program should produce a transfer data set to update the municipality's information. Only information regarding buildings belonging to the owner would be included in this update.

5 MAINTENANCE TRACKING

This section is used to keep records of all maintenance performed on the buildings. It allows the user to schedule preventive or routine maintenance tasks, as well as issue work orders for repairs.

5.1 Work orders

The maintenance tracking sub-system is used to keep building-level and unit-level records on maintenance performed. When a complaint is received from a tenant, a work order is issued that specifies what tasks need to be done. Work orders may also be issued in response to building inspections or preventive maintenance checks.

5.1.1 Issuing work order

Maintenance information is stored in the form of a **work order**. The first section of the work order contains the following information about the problem:

- **Ë** City code*
- **Ë** Cadastral no.*
- **Ë** Building description no. (…íslo pop.)
- **Ë** Supplemental building description code
- **Ë** Work order number (system-generated sequential number)
- **!** Unit no. (if applicable)
- **"** Work order date (date of complaint)
- **!** Date work promised
- **!** Time promised (character field: 11:00am, afternoon, evening, etc.)
- **!** Work order type (preventive maintenance, regular, or emergency)
- **"** Name of complainant
- **"** Telephone number of complainant
- **!** Type of problem*
- **!** Specific nature of complaint (character field)
- **!** ID number of employee assigned to work (from §3.3)
- **!** Date work completed
- **!** Completion status (character field)
- **!** Time required for work
- **!** Total cost of work (kcs)
- **"** ID of person entering work order
- **!** Memo

The second section of the work order describes the specific tasks to be completed. Tasks may be chosen from a list of pre-defined tasks (§5.2) or may be entered directly by the user:

- **Ë** City code*
- **Ë** Cadastral no.*
- **Ë** Building description no. (…íslo pop.)
- **Ë** Supplemental building description code
- **Ë** Work order number

Ë Task number (system generated: 1-5)
! Description of task
! Time required to complete task
! Vendor ID (from §3.2, if applicable)
! Payment to vendor (if applicable)
! Charge to tenant (if applicable)

Finally, the third part of the work order contains information on the supplies needed for the work. If task is pre-defined, then the list of needed supplies should be copied from the task definition (§5.2), with price information coming from the supply inventory (§4.2). Otherwise, the supplies are chosen from the supply inventory (§4.2) or entered directly:

Ë City code*
Ë Cadastral no.*
Ë Building description no. (…íslo pop.)
Ë Supplemental building description code
Ë Work order number
Ë Supply number (system-generated: 1-5)
! Type of supply (from §4.2)
! Quantity
! Unit of measure (pieces, kg, etc.)
! Price per unit
! Cost of supplies
! Status (in-stock or purchase)

When the work order is issued, estimates are entered for the time and cost entries. These estimates will be replaced by the actual time used and cost of repairs when the work order is completed.

Once all the information is entered, the work order is printed. The printout should contain all of the above information, plus spaces for entering the actual time spent and actual payments to vendors for the work. The bottom of the form should have a place for the employee to write the date the work was completed, the completion status, and his or her signature, certifying that the work was completed.

5.1.2 Revise work order

The user should be able to edit information in existing work orders.

5.1.3 File completed work order

This function allows the user to register the work order as having been completed. The program should call up the work order and ask the user to enter the time required for each task, payments made to vendors, the actual number and types of supplies used, the date the work was completed, the charge to the tenant, and the completion status (for example: work completed, no action taken, parts unavailable, could not enter unit, etc.)

Any supplies used should be subtracted automatically from the inventory (§4.2). Payments

made to vendor should be credited to the appropriate expense account and debited from the cash account when payment is made. Charges to the tenant should be added to the tenant's account (§3.4).

5.2 Pre-defined tasks

The program should facilitate entering common tasks that have been pre-defined by the user (for example: change a broken light bulb or repair a leaking faucet). The user specifies the description of the task, the time needed, vendor ID and payments (if any), up to five (5) supplies needed for the task, and charge to the tenant for the service (if any):

- **Ë** Pre-defined task number (system-generated sequential number)
- **!** Description of task
- **!** Time required to complete task
- **!** Vendor ID (from §3.2, if applicable)
- **!** Payment to vendor (if applicable)
- **!** Type of supply #1 (from §4.2)
- **!** Quantity of supply #1
- **!** Type of supply #2 (from §4.2)
- **!** Quantity of supply #2
- **!** Type of supply #3 (from §4.2)
- **!** Quantity of supply #3
- **!** Type of supply #4 (from §4.2)
- **!** Quantity of supply #4
- **!** Type of supply #5 (from §4.2)
- **!** Quantity of supply #5
- **!** Standard charge to tenant for service

When entering a work order, the user may pull up a menu of these predefined tasks and select one. The information on the task is then copied into the appropriate sections of the work order.

5.3 Preventive maintenance schedule

In addition to complaint or problem generated maintenance, the software should also help the user schedule preventive maintenance tasks, such as inspecting heating ducts or testing fire equipment.

5.3.1 Define a preventive maintenance task

This section allows the user to define preventive maintenance (PM) tasks for buildings or units:

- **Ë** PM task number (system-generated sequential number)
- **!** Description of task
- **!** Estimated time to complete

- Frequency (months)
- For building or all units
- Memo

The last field indicates whether the task needs to be performed only once for the entire building or separately for each individual unit.

5.3.2 Preventive maintenance schedule

The program keeps a preventive maintenance schedule as follows:

- City code*
- Cadastral no.*
- Building description no. (…íslo pop.)
- Supplemental building description code
- Unit no. (for unit-level tasks)
- PM task number
- Date task last performed
- Person performing task (character field)
- Completion status (No problems found, Minor problem corrected, Work order issued)
- Next scheduled date to perform task
- Memo

The user adds a PM task to the schedule for the building by choosing a task from the list in §5.3.1. If the task is a building-level task, one new entry is added to the schedule. If a unit-level task, the user should be given the option of adding a task for one unit only or for all units in the building. The user should also be able to edit the information in the schedule.

5.3.3 Completion of PM tasks

This function allows the user to register when PM tasks have been completed. The user can search through the PM task schedule (§5.3.2) and choose the task that was completed. The program prompts the user for the date the task was performed, the name of the person completing the task, and the completion status. The program then automatically updates the next date for performing the task by using the frequency provided in the task definition (§5.3.1).

If the completion status was "issue work order," then the program should ask the user if he wishes to enter a work order at this time. If the user says "yes," then the program allows the user to enter the work order information in §5.1.1.

5.4 Reports

The maintenance tracking sub-system should provide the following reports:

Listing of outstanding work orders: This is a list of all work orders that have not been completed.

Completed work orders: This report lists all work orders that have been completed during the range of dates specified by the user. The report should show the date the work was completed, the tasks performed, and the completion status.

Complaint report: The program should be able to produce a report of the number complaints of each type made during the time period specified by the user.

Maintenance cost: This report gives the number of hours spent, approximate cost of employee time (calculated by multiplying hours spent by the employee's hourly wage given in §3.3), cost of materials used (quantity of material times the price per unit in §4.1), payment to vendor, charges made to the tenants, and total cost for each type of complaint during the time period specified by the user.

Schedule of preventive maintenance tasks: This is a list of all PM tasks scheduled to be completed during the time period specified by the user.

Completed preventive maintenance tasks: This is a list of all PM tasks performed during the time period specified by the user. The report should give the date the work was completed, the person completing the task, and the completion status.

Each of the above reports should be available for individual buildings, all buildings, or for a group of selected buildings.

5.5 Differences between the programs

Detailed maintenance data is important to the owner as well as the manager of the building. Therefore, the manager's program should be capable of creating a transfer files containing all of the data described in this section. This file would be read by the municipality's program to update the data. The program should also be capable of producing a report that compares the maintenance data for the different management companies employed by the municipality.

6 REFERENCE DOCUMENTS

This section of the program allows the managers and owners to have easy access to documents that are useful to them. Such documents would include the text of laws or decrees affecting housing, standard management contracts or tenant leases, delinquency letters, etc.

The reference sub-system should contain a menu with the following commands:

- Edit a document
- Print a document
- Add a document
- Delete a document
- Format output

The edit command allows the user to edit those documents already in the system. Choosing this command should cause a list of documents to appear, each with a brief description. Once the user selects a document from the list, the program loads the document into the editor. The document editor should support basic editing functions (delete, insert, backspace, word wrapping, tabs, import ASCII file, etc.). The print command prints a document. The add command would allow the user to add a new document to those that already exist. A brief description of document would be entered, and then the user would be able to add the text in the edit function. Delete a document would delete an existing document.

The format output function would give the program information on how to print the documents:

- Size of page (number of lines and columns)
- Left, right, top and bottom margins
- A character string to be sent to the printer before printing document

7 TECHNICAL SPECIFICATIONS FOR SOFTWARE

The following are additional notes on the technical requirements for the software.

7.1 General features of the software

Although this specification provides structures of data files, this information is given mainly to describe to the programmers the way the data elements should relate to each other. It is not meant to imply that the programmers must use precisely the file structures included in this specification. The programmers should feel free to modify the organization of the data if these modifications will result in faster access to the data or better use of memory or storage space, provided, however, that the intended functionality of the software is retained.

The software should operate on IBM compatible PC's using the DOS operating system (version 3.30 or later). The software may require conventional memory up to 640 kilobytes, but it may not require any additional expanded or extended memory. (It will be considered an advantage to the firm submitting the tender, however, if its software can make use of expanded or extended memory to improve performance.) The software should not require the computer to have a math co-processor, but should make use of one if it is present.

The software should be network compatible, that is, it should be able to operate on a local area network (LAN) with multiple users having access to the program and the data files.

The program should be capable of importing and exporting all files (§1) in either ASCII or dBase-compatible file formats.

The program should be user-friendly and menu driven. It should have on-line, context sensitive help available by pressing a function key or selecting a menu command. All program text (commands, menus, messages, prompts, output, etc.) should be in the Czech language.

The user should be able to execute DOS commands without actually exiting the program (DOS shelling).

The program should be password-protected, requiring users to type a user ID and password before being able to access the programs functions. The program should recognize different levels of users, as indicated in Figure 7.1. (Access levels not defined in Figure 7.1 are reserved for future use).

Figure 7.1 - Levels of user access to software

User level	Program access
9	**Supervisor:** Full access to all program functions. The only user level who can add or delete program users, modify user passwords, and change user levels.
7	**Accountant:** Access to all program functions, including the general ledger (3.8), and payroll data (3.3).
5	**Technician:** Access to all functions, except general ledger (3.8) and payroll (3.3).
3	**Data entry operator:** Access to housing database functions only.

The program should have a function for backing up the data onto diskettes.

Within the constraints mentioned above, the programmers may develop the software in whatever environment they choose (e.g. FoxPro, dBase, C, Pascal). The software must be delivered as a "stand-alone" program, with all programs necessary to run the software from the DOS prompt.

7.2 User data

The program should store data on the user of an individual copy of the program, whether owner or manager. Some of these data will be used on report headings to identify the source of the reports. Information will include: name of user (person or firm), address, telephone number, fax number, ICO number.

7.3 Transfer of data between remote and central sites

Since it is the case that a management firm may have several branch offices that must exchange data with a central office, the software should have a function for transfering data from computers at remote sites to a central office computer. This function would create transfer data files containing all data that have been entered or modified since the last transfer took place (or since another date specified by the user). These files could then be loaded onto the central office computer via diskette or by transmitting them over phone lines via modem. (The HMS software is not expected to provide support for modem communication.) A corresponding function on the other system would then read the data in the transfer files and update the information in its database.

The transfer files should be numbered so that data from previous transfers is not confused with the current set of transfer files. For example, XDOMY.013 could be the 13th transfer file of the building data. A special "control file" should contain information about the transfer: the date the transfer files were created, the identifying information about user creating the transfer files in §7.2,

the time period of the data included in transfer, and the number of transfer files produced. The user should be able to inspect the information in this file in order to identify the transfer.

7.4 Transfer of data between municipality's and manager's program

It has been discussed throughout that the municipality's program should be able to receive data from the building managers responsible for its buildings. A single function in the program should handle the creation of transfer files containing all data required to update the municipality's system. This function will be similar to that discussed in §7.2, except that it should transfer data for only those buildings belonging to a particular municipality.

Before creating the transfer files, the manager will choose an owner from the list of owners given in §2.8. Only the data for buildings belonging to this owner will be included in the transfer. As with the transfer described in §7.3, the transfer files should be numbered so that data from previous transfers is not confused with the current set of transfer files. A special "control file" should contain information about the transfer: the date the transfer files were created, the identifying information about user creating the transfer files in §7.2, the time period of the data included in transfer, and the number of transfer files produced. The owner should be able to inspect the information in this file in order to identify the transfer.

7.5 Selection of buildings for reports

For all of the report's described above, the user should be able to generate output for either: (1) a single building, (2) a set of buildings, or (3) all buildings. For example, the user should be able to select an individual building by searching through a list sorted by city, cadastre, and description no., or by city and street address. Housing managers should be able to create reports for all buildings belonging to a particular owner, and municipalities should be able to create reports for all buildings managed by a particular management firm. Finally, the user should have the option of creating reports for all buildings in the database.

The report headings should include the name and address of the user from §7.2, the date the report was created, page numbers, and a report title.

The report generation system should allow the user to specify the paper size (lines and columns) that the report is to be printed on, the size of the top, bottom, left, and right margins, and a printer setup-string (that is, a set of ASCII characters that will be sent prior to each output to initialize the printer). The program should allow the user to direct output to a printer port (LPT1 or LPT2) or a disk file.